UNIVERSAL LOAD ADDRESS/VALUE PREDICTION SCHEME

Prior Foreign Application

[0001] This application claims priority from European patent application number 00111339.8, filed May 26, 2000, which is hereby incorporated herein by reference in its entirety.

Technical Field

[0002] The present invention relates to performance improvements in superscalar computer systems. In particular, it relates to an improved method and system for hybride address prediction.

Background of the Invention

[0003] To achieve higher performance most microprocessors are designed as superscalar processors having multiple execution units. The idea behind this concept is to increase instruction level parallelism further referred to herein as ILP. Because most instructions show dependencies which would lead to stalls in the processor's pipeline(s) until the dependency is resolved register renaming in combination with out-of-order execution allows improvements of ILP. Nonetheless a lot of dependencies still remain and prevent multiple instructions from being executed in parallel which leads to bubbles in the pipeline.

[0004] To increase efficiency and to overcome the bubbles in the pipeline load address or value prediction can help to avoid pipeline stalls, because even dependent instructions can be executed using speculatively calculated data. If it turns out that the predicted value was wrong the corresponding instructions must be re-executed which represents a performance reducing penalty.

[0005] To reduce the penalty for mispredicted values it is a) necessary to provide best possible load address/value prediction and b) necessary to determine the instructions whose operands can be predicted with high confidence and which cause low penalty even if the predicted address/value was wrong.

[0006] In particular, prior art value prediction can be separated into three categories: Load address prediction, prediction of source register values and prediction of target register values.

[0007] The simplest algorithm used in prior art value predictors is based on the assumption that the contents of memory locations and registers remains mostly unchanged. So, an appropriate prediction scheme is simply to predict the last value. The so-called last value predictor, further referred to herein as LVP, too, is exemplarily depicted in Fig. 1 and comprises a table 10 which is addressed by hashing 12 of the instruction address with each entry consisting of a tag field 14 and a field 16 for the last value.

[0008] The table is most likely organized as n-way set associative (e.g. n=4). If a match is found by finding out that the tag field matches the instruction address then the corresponding last value from this table entry is used for prediction. If there is no match a new entry is made replacing the Least Recently Used (LRU) table entry determined by a so-called LRU algorithm.

[0009] Anyway, the predictor is updated each time with the correct value, if it is known/confirmed.

[0010] Another prior art scheme is a simple extension of the LVP, as it is depicted in Fig. 2.

[0011] Two additional fields, the stride field 20 and a status field 22 are added to each table entry. The idea behind this predictor is that often memory contents are changed by a certain delta value, i.e. a stride. Thus, the next predicted value can be calculated by simply adding the stride to the last value. The status field is used to determine whether the predictor should predict the last value or the last value increased by a certain stride. So the stride predictor further referred to herein as SP is involved only if a certain stride could be found and confirmed whereby the status field keeps track of that.

[0012] If the stride predictor fails after some successful predictions it will switch back to last value prediction (switching the status field back to LVP) unless a new stride is found and confirmed.

[0013] The stride predictor, further abbreviated herein as SP is updated every time with the most current value. If the stride changes it is used only if the new stride is confirmed, i.e. when the same stride is found the next time again.

[0014] It should be noted that such a confirmation is advantageously done when the same stride re-occurs at least twice subsequent to each other.

[0015] Despite that the LVP and SP can achieve correct prediction rates of up to more than 50%. For certain cases there are still some instructions which alter the contents of memory locations according to a particular pattern which is repeated several times. So values can be predicted out of such a context and a so-called context predictor, further referred to herein as CP, too, has been proposed.

[0016] Whereas the SP is an extension of the LVP the context predictor is based on a two-table lookup and thus consists of two tables as is illustrated in Fig. 3.

[0017] The entries in the first table 30, which is organized n-way associative, each comprise a tag field 14, several (e.g. four) last value fields 31a-31d, a LRU info field 32 and a value history pattern field 33. An entry is selected via hashing 12 of e.g. an instruction address. If no match is found a new entry is added to the table replacing the least recently used table entry according to the LRU info. Said step of adding a new entry in particular

comprises to write the tag info -e.g. the instruction address- in the tag field, to write the current result produced by the instruction in one of the value fields 31a - 31d, and initializing the value history pattern stored in fields 33.

[0018] The value history pattern describes the history of the last several (e.g. six) values of the selected memory location used in a series whereby each of the value fields 31a - 31d is identified by a two bit pattern. '00' refers to the value stored in the value field 0, '01' refers to the value stored in the value field 1, etc.. For example, if the six most recently used values of a certain instruction were placed in value fields 0,1,2,0,3,2 the corresponding value history pattern (VHP) is '00 01 10 00 11 10'. The LRU field stored in each table entry determines which value field is overwritten/ replaced if a new value is detected for that instruction.

[0019] The two-table lookup is done by using the VHP (e.g. a 12bit pattern) as an address to select an entry in the second, the so-called pattern history table 34, further referred to herein as PHT, too. Exemplarily, said second PHT table may have a number of 4K entries in conjunction with the 12bit pattern used to address this table.

[0020] An entry in this table comprises four saturating 4bit counters 35a to 35d. These counters represent each value field 31a to 31d in the first table 30. The counter with the highest value and with a count higher than a

threshold value selects the appropriate last value stored in the first table. The counters in the PHT are updated according to the current value, i.e. the corresponding counter is increased by a certain number (e.g., by 3) whereas the other counters are decreased (i.e. by 1). The counters saturate (e.g. by 0 resp. 12), and the threshold value is chosen (e.g. To be 6) to determine whether a prediction can be made or not.

[0021] The second update procedure comprises to update the VHP 33: The VHP 33 is shifted left two bits and the vacant two bits on the right are filled with the bit pattern corresponding to the current value. If the value was not already stored in one of the 'last value fields' the current value replaces the least recently used last value stored in one of the four value slots and the corresponding two-bit pattern is placed into the VHP 33.

[0022] Whereas such a context predictor predicts certain repeating patterns of values -here patterns consisting of up to four different values- it is not effectively predicting strides or last values. Therefore, best prediction can be achieved by combining the CP with the LVP/SP. This 'combined' predictor is often called a hybride predictor (HP). It uses a certain switching scheme to select the predictor of choice to achieve best reliability.

[0023] An advantage of the hybride predictor is that is saves latch counts for using the SP for last value and stride predictions. The major drawback, however, is the

complex underlying switching scheme which is necessary in prior art to decide whether to use the LVP or the SP or the CP. According to prior art it is preferred to start the prediction always with the LVP. If the LVP is not successful, but a stride could be found and confirmed, then the SP is invoked. If no stride could be determined, then the CP is initialized and starts collecting and confirming the pattern -assuming that there is a certain pattern of values.

[0024] If the pattern stabilizes, i.e., the counters in PHT 33 reach the threshold value, predictions can be made out of context. If the predictor fails, the switching scheme re-enables the LVP. The disadvantage is, however, that the context predictor invocation is rather inefficient because it takes rather long until the CP is really used because prior to issueing a context prediction the data must be collected which are the basis for a reliable CP.

Summary of the Invention

[0025] It is thus an objective of the present invention to provide for a prediction scheme which supports value prediction, stride prediction and context prediction with reduced storage requirements and with a better performance when switching between said different kinds of predictors.

[0026] This objective of the invention is achieved by the features stated in enclosed independent claims. Further

advantageous arrangements and embodiments of the invention are set forth in the respective subclaims.

[0027] The present invention discloses a new load address/value prediction scheme which combines the advantages of the three prior art prediction schemes LVP, SP, and CP described above.

[0028] Said new scheme for value prediction allows to predict last values, strides as well as values out of context without the use of a sophisticated switching scheme between several predictors. Thus, a quite 'universal' prediction (UP) scheme is disclosed which is based on the two-table lookup mechanism of the context predictor but which deals with differences between subsequent values stored in a certain memory location.

[0029] The inventional prediction system collects patterns of deltas of subsequent values instead of the values itself. Thus, a LVP can be achieved by predicting a 'pattern' of just one stride which is zero. A stride predictor uses a pattern consisting of just one (constant) stride. And a certain pattern of values is now modelled by recording the pattern of deltas between the values and adding the deltas to the last value.

[0030] As the context prediction is based now on the deltas, i.e., the differences between some values, the predictor is also capable of predicting values which show a certain pattern of changes. This is thus more general than

just recording a certain pattern of values. The main advantage of the inventional predictor concepts is that it involves the switching scheme inherently, i.e., if a certain counter reaches a hit-threshold value, the prediction out of context including stride prediction as well as last value prediction is started.

[0031] According to a preferred embodiment thereof the default and initial prediction method is LVP by using a stride = 0. This can be achieved by initializing the corresponding counter to the threshold value. If the value is not predictable at all, this counter will be decreased below the threshold and the new status 'not predictable' will be recognized and can be issued. This is a remarkable advantage compared to prior art because the performance penalty due to a misprediction recovery can be remarkably higher than waiting until the dependency is resolved and the result is calculated in an ordinary manner.

[0032] If the last value prediction or the stride prediction is correct the predictor will immediately start using these prediction schemes. If no stride could be found but a pattern can be detected instead, the predictor has already begun with collecting and confirming this pattern and will start using the context prediction mechanism as soon as possible.

[0033] The predictor saves thus array counts, because the strides stored in the stride fields may have a restricted number of bits compared to the last value stored in the CP.

This is true despite the fact that the last value must be stored in an additional field in each entry. Assuming that the values to predict are 64 bits wide and that a stride field consisting of 16 bits is sufficient, four stride fields and the last value field together will consume 128 bits, whereas the CP with four last values stored in each entry will consume as much as 256 bits.

[0034] Advantageously, the number of stride fields is greater than 3 and smaller than 7 for being applied in today's modern computer architectures.

Brief Description of the Drawings

[0035] The present invention is illustrated by way of example and is not limited by the shape of the figures of the accompanying drawings in which:

[0036] Fig. 1 is a schematic block diagram showing the essential components used in a prior art last value predictor,

[0037] Fig. 2 is a schematic block diagram showing the essential components used in a prior art stride predictor,

[0038] Fig. 3 is a schematic block diagram showing the essential components used in a prior art context predictor,

[0039] Fig. 4 is a schematic block diagram showing the essential components used in a hybride predictor according to a preferred embodiment of the present invention, and

[0040] Fig. 5 is a block diagram showing basic steps and control during operation of setup and update procedure of said preferred embodiment of the present invention shown in Fig. 4, and

[0041] Fig. 6 is a block diagram showing basic steps and control during operation of the prediction procedure of said preferred embodiment of the present invention shown in Fig. 4.

Best Mode for Carrying Out the Invention

[0042] With general reference to the figures and with special reference now to Fig. 4 the essential components used in a hybride predictor according to a preferred embodiment of the present invention which is referred to herein under as 'universal predictor' (UP) are described in more detail next below and applying for instruction address prediction for 64bit addresses.

[0043] The universal predictor is a two-level predictor comprising two tables 40 and 44. The entries in the first table 40, which is organized fourfold associative, comprise: a (prior art) tag field 14, 32 bit long, a LRU info field

32, 6 bit long, dependent of the number of stride fields in use, a last value field 42, 64 bit long, four stride fields 41a to 41d, each 16 bit long, and a stride history pattern (SHP) field 43, 6 time 2 bits = 12 bits long.

[0044] An entry of said table 40 is selected via hashing of the instruction address. If no match is found a new entry is added to the table replacing the least recently used table according to another LRU info. This is a 6bit pattern for each hashing address which keeps track of the least recently used table entry of the fourfold set-associative organized first table 40.

[0045] When a new instruction occurs the first time during operation no stride will be known for it and a new entry must added which comprises to write the tag info, i.e., the instruction address, into the tag field 14, to write the current value in the last value field 42 and to write stride = 0 into one, e.g., the first of the four stride fields 41a,..41d, and initializing the stride history pattern by e.g. '00 00 00 00 00 00' if stride = 0 is written into the first stride field. Thus, for the next time, at most a stride=0, i.e., the last value can be predicted. When a stride not equal 0 turns out to be true, than some delta exists, LVP turns out not to be adequate and said delta can be taken as stride for future prediction replacing the former stride=0 in the stride-0 field 41a.

[0046] The stride history pattern describes the history of the last six strides used in series where each stride is

identified by a two bit pattern, e.g., '00' for the stride placed in the stride filed 0, '01' for the stride placed in stride field 1, and so on. When for example the six recently used strides were placed in stride fields 0,1,1,0,3,2 then the stride history pattern (SHP) would be 00 01 01 00 11 10.

[0047] A second LRU info stored in the LRU info field 32 of each table entry determines which stride in the stride fields has to be replaced if more than 4 strides are needed and the least recently used stride is replaced.

[0048] The two table lookup is then done using the stride history pattern SHP (a 12 bit pattern) as an address to select an entry in a second, so-called pattern history table 44 (PHT) having 4 K entries. An entry in this table comprises four saturating 4bit counters. Each counter 45a..45d is associated to a respective stride field 41a..41d in the first table 40. The counter with the highest value and with a count higher than a particular predetermined threshold value selects the appropiate stride which is used for the prediction which is then done like in prior art -see the bottom portion of Fig. 3 and 4, but - based uniformly on strides instead of separately evaluating values, strides and value based patterns. The predicted value is calculated by an addition of the selected stride and the last value. If the counter(s) in the PHT 44 are below said treshhold value, then no prediction will be made, thus a status 'not predictable' is granted in the respective cycle.

[0049] Next, the update and initialization procedure of the counters will be described in more detail as it reveals some important aspects of the present invention.

[0050] In order to provide a short setup time of the predictor, the number of requests to a certain table entry until a prediction for the corresponding instruction can be made should be hold as small as possible. Thus, a particular initialization of the predictor is required.

[0051] According to a preferred embodiment of the present invention a prediction will start immediately after a new instruction is stored in the LVP/SP, i.e., the next time the instruction is hit the LVP will predict the last value.

[0052] If the last value is wrong the current difference is stored as a stride and the next time prediction can be made using this stride. Despite that, the predictor will still predict the last value until the stride is confirmed.

[0053] Without a special initialization the inventional concept proposes to predict only if at least one counter in the PHT will exceed a certain threshold value. This means that depending on the counter update procedure - comprising in turn increasing of the correct PHT counter and decreasing the remaining counters - it needs several requests to the predictor until the predictor actually starts the value prediction.

[0054] In particular, when a new instruction is found and a new entry is written into the first table the current value is placed in the last value field, stride 0 into the str0 field and the LRU is initialized so that the next stride is written into str0 replacing stride 0. The SHP is initialized with the pattern '00 00 00 00 00 00' which describes a valid history for a last value/stride predictor which always uses the stride stored in str0. This pattern is unique for a LVP/SP and the corresponding counters in the PHT must be set appropriate to ensure that the prediction will use str0, i.e. the first counter is set to a value well above the threshold (e.g. to the maximum value 12) and the other counters to a value well below the threshold (e.g. = 0).

[0055] Assuming that every stride field (str0, str1, str2 or str3) can be used in LVP/SP prediction the corresponding SHP ("00 00 00 00 00 00", "01 01 01 01 01 01", "10 10 10 10 10 10" or '11 11 11 11 11 11") address certain counters in the PHT which can be initialized (and even fixed) appropriate: If the stride used for prediction is stored in field str2, the second counter of entry '101010101010' in the PHT is preset to a value well above the threshold the remaing counters to values well below the threshold value. Accordingly, the following PHT entries can be preset (and even fixed) to the following counter values:

SHP = PHT-address	**PHT-cnt0**	**PHT-cnt1**	**PHT-cnt2**	**PHT-cnt3**
00 00 00 00 00 00	12	0	0	0
01 01 01 01 01 01	0	12	0	0
10 10 10 10 10 10	0	0	12	0
11 11 11 11 11 11	0	0	0	12

[0056] Thus, the step of adding a new instruction into the proposed predictor will take advantageously the following steps:

1. Step: write a new entry in the first table upon detection of new instruction:

   The new entry is addressed via the hashing function. The SHP is initialized with a pattern for LVP/SP using sr0 ('00 00 00 00 00 00'). Thus, the SHP points into PHT entry '000000000000' with its counters set to: c0=12 (max), c1=c2=c3=0 (min).

   Thus, the c0 counter 45a points to str0 field 41a which can be used in the next cycle for a last value prediction.

2. Step: applies if a stride not equal 0 is found: As the stride field 41a sr0 is used for prediction, SHP remains '00 00 00 00 00 00'

   A stride of x is written into stride field 41a-sr0, whereas x is the difference between the current value and the last value. The next prediction then corresponds to (lastvalue + x). To ensure that the stride is written into sr0 - replacing the stride 0 - the LRU must be initialized accordingly as described previously.

3. Step: if no single stride is found:

The prediction still uses sr0, but no stride is stored in the empty stride field. The SHP is changed, depending on the stride field used: if str1 field 41b is used to store the new delta, the corresponding SHP will be '01 00 00 00 00 00'.

[0057] The corresponding counters in the PHT remain unchanged, i.e., they may have the initial values somehow below the threshold value, e.g. 3 with a threshold of 6, or the values which were already adjusted by another instruction which obeys the same stride history pattern.

[0058] If all corresponding counters in the PHT are below the threshold value no prediction will be made the next time. If a certain stride pattern is detected and confirmed, i.e., at least one counter exceeds the threshold value, predictions are made by using the stride field specified by said PHT counter with the highest value.

[0059] In this way an immediate response of the inventional prediction method to the neutral start conditions as well as to the creation of new table entries can be achieved.

[0060] With reference now to Fig. 5 the basic steps in the control flow during the setup of the counters and the update procedures of the relevant fields in tables 40 and 44 are described in more detail:

[0061] In a first step 510 - when the program is started- all counters are initiated, i.e. setup - advantageously according to the scheme given above.

[0062] When a result is available from a newly completed instruction, see yes branch of decision 520, it is checked, decision 530, if the same instruction can be identified to be present in table 40. Thus, the tag field 14 in table 40 is checked and the tag compared with the instruction address. As long as no result is available, - see the no branch - control is fed back to repeat the check 520.

[0063] In the no-branch of decision 530, i.e., when no matching entry was found said current instruction is installed in the first table 40, block 540: In particular, the tag field 14 is written, the SHP field 43 is setup as well as the LRU field 32, and a stride of 0 is written into stride field 41a of the respective new entry in table 40.

[0064] Else, yes branch of 530, the current stride is calculated by subtracting the last value from the current result, step 550.

[0065] Then, it is checked if the same stride can be found in one of the stride fields 41a,.. 41d, decision 555.

[0066] If not - in the no branch of 555 - the current stride is stored into the respective stride field which is specified by the value store in the LRU field 32 and said LRU field is updated, block 560.

[0067] In the yes-branch of 555 a current stride was already stored in one of the stride fields. Now, as well as after performing block 560, the corresponding PHT counters 45a,.. 45d are updated, block 565, by increasing the correct counter by 3 and decrementing the other counters by 1. It should be noted that the respective entry in table 44 is addressed by the SHP in field 43.

[0068] Then, as well as after performing block 540 the new stride history pattern is calculated as described further above, step 570. In particular, the SHP field 43 is shifted left by two bits and the vacant bits on the right are replaced by the bit pattern corresponding to the current correct stride. If this stride is not found, the current stride is written for replacing the least recently used stride field, and the corresponding 2bit pattern is placed in the SHP 43.

[0069] Finally, the result is stored in the last value field 42, step 575 and control is fed back to decision 520 in order to process the next instruction upon its completion.

[0070] With reference now to Fig. 6 the prediction procedure is described in more detail. It should be noted that - in said preferred embodiment - the update/setup procedures and the now described prediction procedure are implemented as independently running processes which access the same hardware arrangement by respective write (Fig. 5) and read accesses (Fig. 6), respectively.

[0071] An arbitrary instruction is treated according to the following control scheme:

[0072] In a step 610, the instruction is first decoded. Then, it is checked in a decision 620, if the same instruction can be identified to be present in table 40. Thus, the instruction address is compared with the tag stored in tag field 14 in table 40.

[0073] If no matching instruction is found, no prediction is possible, block 630, and the status 'not predictable' can advantageously be signalled, i.e., issued in order to prevent a misprediction, see step 635. Then the control is fed back for decoding the next instruction, to step 610, again.

[0074] Otherwise, yes-branch of decision 620, after tag-hit, the stride history pattern is read from the first table 40, field 43, step 640. This pattern is used for selecting a respective matching entry in the second table 44 in order to evaluate and select the counter values, step 650.

[0075] Thus, the counters and the corresponding patterns can be read and evaluated, in particular, if any counter's current count is above a predetermined threshold value, of e.g., 6, decision 670.

[0076] If, in the yes-branch of 670, a counter has a count of greater than the threshold value of 6 the respective prediction can automatically be undertaken by

selecting the highest counter, step 660. This is a remarkable advantage compared to prior art which needs a complicated switching scheme in order to change from LVP to SP, and in particular from SP to CP.

[0077] Then, in a step 690 the current predicting value is calculated by adding the last value to the stride selected by the highest counter. Then, control is again fed back to step 610.

[0078] In the foregoing description the invention has been described with reference to a specific exemplary embodiment thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are accordingly to be regarded as illustrative rather than in a restrictive sense.

[0079] In particular, the dimensions of the fields given in the above exemplary embodiment may be varied as required for the respective processor architecture in use.

[0080] Further, the present invention can be included in an article of manufacture (e.g., one or more computer program products) having, for instance, computer usable media. The media has embodied therein, for instance, computer readable program code means for providing and facilitating the capabilities of the present invention. The

article of manufacture can be included as a part of a computer system or sold separately.

[0081] Additionally, at least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform the capabilities of the present invention can be provided.

[0082] The flow diagrams depicted herein are just examples. There may be many variations to these diagrams or the steps (or operations) described therein without departing from the spirit of the invention. For instance, the steps may be performed in a differing order, or steps may be added, deleted or modified. All of these variations are considered a part of the claimed invention.